Environmental feature extraction and comparison using a rotating Infra-Red sensor

(Bachelor Thesis)

Marc Volger, s1632531, M.Volger@student.rug.nl,
Sjoerd de Jong*, Herman Kloosterman†

22 juli 2010

Abstract

For an autonomous robot it can be quite useful to locate its own position within its environment, which in the case of a vacuum cleaning robot is often a living room. Certain features from this environment can help with the self localization, like for example corners. In this bachelor project we have tried to extract such features from the data of a rotating infra-red sensor, when positioned in different places relative to the features. The recording of the data is done in a laboratory at Philips Drachten resembling a common living room. After the features have been extracted, they are described by descriptors. This way each recording of the environment gets its own descriptor. The descriptors are matched with each other by means of the Levenshtein distance. The wellness of the matches between the descriptors, combined with an internal map of the environment, can help us say something useful about the probable location of the robot within that environment.

1 Introduction

This research fits within the frame of Artificial Intelligence because it is research on the use of a sensor by an autonomous robot. Handling and processing sensor data covers a great part of the ability to the well-functioning of such an autonomous vacuum cleaning robot. Feature detection from the sensor data is again a substantial part of the processing, because through that the robot can know what is in his environment and react on that knowledge.

*University of Groningen, Department of Artificial Intelligence
†PHILPS Drachten

At Philips Drachten there has been done some research in the field of sensors for use on autonomous vacuum cleaner robots which operate in living rooms. Infra-red sensors with triangulation [1] are used in that research as well as the relative new Time-of-Flight [2] sensors. The production and acquisition of an infra-red sensor is considerably cheaper than a Time-of-Flight sensor and therefore it is much more plausible that they will be used on the robotic vacuum cleaners in the near future. There is a trade-off between costs of the sensor and the quality of the data that it captures. In this research we chose to work with the infra-red sensor. This sensor does capture much less information then the Time-of-Flight sensor about its environment. Therefore it shall be used for tasks that require much less computation on the data, like for example corner detection. In [3] and [4] there is a good argumentation why for example an ultrasonic sensor is not suitable for our research. Among other things it is too noisy, expensive and annoying for the user. The advantages of the infra-red sensor we use are that it is of low cost, low weight and can be spun to record its total environment. There is also the prospect of the sensor appearing on the autonomous vacuum cleaning robots in the near future. These were mainly the reasons why we used a small infra-red sensor, despite of its low data quality in comparison with the Time-of-Flight sensor, for this experimental research at Philips Drachten. Any results of the research may be useful for Philips in the future.

The purpose of this experimental research is to detect various features of the environment of the robot, which can help the robot to locate itself in

that environment. The research in [5] had the same purpose as ours, and they have used a quite similar infra-red sensor as well. Their results were good and they made some positive conclusions. In our research we will try to extract the features from a dataset created with the rotating infra-red sensor. The sensor will be placed relative to certain known features in different positions. The features will be an inner corner, an outer corner and a small table which consist of inner and outer corner features. Then when the features are extracted, they will be described by so called descriptors, which are compared with each other by calculating their match. When you can detect features in your environment and you know where the features are to be found in the environment, by means of a map or some sort, you can use the information on where the features have been seen to localize yourself. The question whether the features can be detected and how well this can be done is central in this research.

The features should be recognizable an repeatable, which holds that they can always be found in the environment at different moments if they are present. Therefore they should also be robust and have viewpoint invariance. That way, they will match from every position you will see the feature. The features should also posses match quality and noise insensitivity.

In our research we scan the environment and will create a datamap out of which we will try to extract features. The datamap will be shown in a graphic representation, which will show different environments and different features. This is the same in the research in [3] where the authors also represent the environment graphically. In their research however, they have used a robot with 24 sensors which were in fixed positions on the body of the robot. We on the other hand use a rotating sensor that gives an output every 2 degrees it is turned, which results in 180 recordings of the sensors environment in a full rotation. This is an improvement on the earlier named research. Physically our sensor is implemented in a way that is quite similar to the set-up in the research of [4]. Both that and our research have a set-up in which the sensor rotates. The physical implementation of the sensor is done by a master student Electrotechnique for Philips.

2 Method

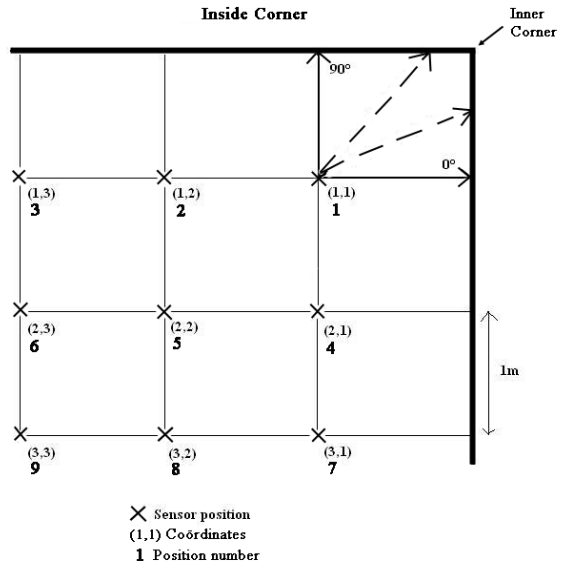
2.1 Experimental setup

For the experiment we have used a laboratory that resembles a real living room to create a realistic environment for a vacuum cleaning robot. We have used this laboratory because the vacuum cleaning robots are mainly used to vacuum living rooms in the real world. In the laboratory the sensor is placed relative to different environmental features. The sensor is set up on a grid of whole meter cordinates. As mentioned above, the features we have used are an inside corner, an outside corner and a small table which consists of the inner and outer corner features. A graphical representation of the setup of the experiment is given in figure 1 and 2. The inside corner feature is shown on the grid in figure 1. The outside corner with a size of one square meter which extends to position number 1 is shown in figure 2. Each of the nine numbered positions have their cordinates on the grid. These numbers will be used throughout the whole paper.

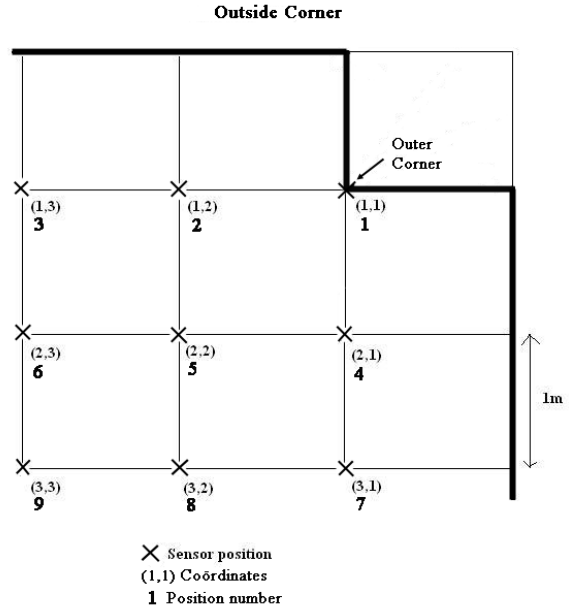
For the inner corner, data is recorded from all the 9 positions on the grid. The outer corner is only recorded from 8 positions since position 1 was not available. The table, which has a size of 50x50x46 centimeter and is placed against the inside corner, is recorded from 6 positions, which were the 1-6 positions. In total there were 23 recordings done. The way we have set up this experiment makes sure that the experiment is repeatable and that the features will be recognizable, assuming our method works correct.

2.2 Dataset

When the sensor is in place, we orient it in the direction of the right hand wall at an angle defined as 0. Then we let the sensor rotate until it has fully seen the feature and passed it. A turning angle of 90 degrees is sufficient (see figure 1). During the rotation, the data that is generated is recorded and stored in a file. This is repeated for every coordinate on the grid for every feature. The data that is generated consist of information with intervals of 2 degrees, which all have their own distance-reading from the sensor.



Figuur 1: Graphical representation of the sensor grid with position numbers and an inside corner.



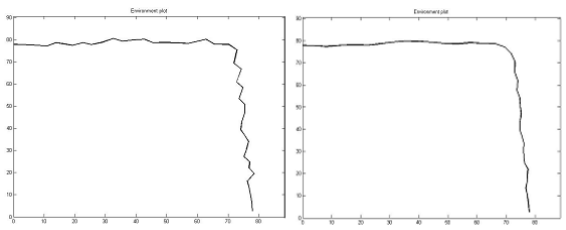
Figuur 2: Graphical representation of the sensor grid with position numbers and an outside corner.

2.3 Data pre-processing

After the data is captured it has to be pre-processed before feature detection can take place. First the 45 readings at 2 degrees interval, that have captured the feature, are extracted from the data file. This way we make certain that we have covered the 90 degrees of view in which the feature is positioned. The polar cordinates form the distances and angles are converted into Cartesian cordinates on an x/y-axes system. We do this to make further calculations on the data easier. Then to make the features have more noise insensitivity, the data is smoothed by a 3-point-moving-average filter over neighboring data points to filter some of the noise from the sensor out of the data. A moving-average filter is a standard filter in Matlab, which is our programming environment. We set the filter to three points, which means that the filter will calculate the sum of the data point itself and its two surrounding data points and then divides it by three. This way you will compute the average of three data points for one data point. Once the filter is done it moves to the next data point, hence the term 3-point-moving-average filter. The effect of the filter is that the data points will be a little more in line then they used to be before the filter. This will also cause the sharp corners in the environment to get a little more blunt. An example of the smoothing of recorded data is shown in figure 3.

2.4 Feature Detection

Our first step in the feature detection is to extract one of the simplest features from the environment, which is a corner. We do this by looking at the second derivative of the data points. First we ob-



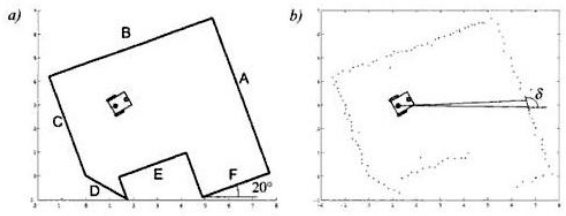
Figuur 3: Left: Graphical representation of the original data taken from position nr 5. Right: Graphical representation of the smoothed data.

tain the first derivative for all of the data points by computing the angle between two consecutive data points. We will put all the obtained angles in a vector to make further calculation easier. The same method of angle calculation is used in [6]. Figure 4 taken from that book shows a graphic representation of a robot in a simple environment (a) and the method to calculate the several angles from the data points that are recorded (b).

From the angle vector mentioned above we again compute the derivative, which shows us the change in angles in the data and is called the second derivative. A significant change in the direction of the angles in the angle vector means that there is something interesting in the environment. If we plot the second derivative we can detect the corner features by a peak in the plot, which occurs at a large change in the angle vector. What we can say about the peaks in the second derivative plot, is that at the angle corresponding to the peak there is a corner. This corner is an inside or an outside corner depending on the peak being positive or negative respectively. In the height and width of the peak there is information of the angle of the corner. We use this information to define what is labeled as a corner and what is not. If the value of the second derivative of a data point exceeds the threshold 10 degrees (or 10 degrees for an outer corner), then it is marked as a nominated corner feature. Then it is checked whether four surrounding data points have a higher second derivative value. The data point with the highest second derivative value is labeled as the actual corner feature. The threshold of 10 degrees was found experimentally to be a safe and effective value.

Line pieces can also be detected within the data points if they are present. We define a line piece as four consecutive data points with angles that differ only within a range of 10 degrees from each other. Then we know with some certainty that the data points are on an approximate straight line.

When the features are detected they can be plotted in the graphical representation of the environment. An example of a graphical representation of the environment with the features plotted in it is shown in figure 5. The representation of an outer corner is shown, with some line piece features, two inner corner features and one outer corner feature in it. The line pieces are shown by dots on their beginning and end.

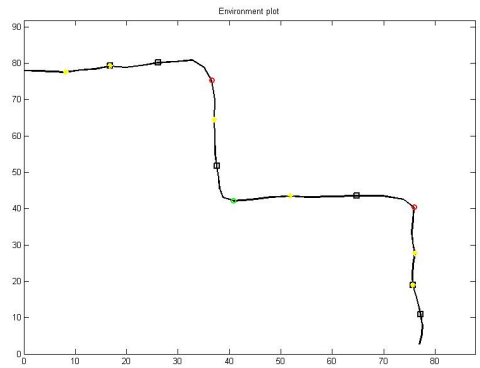


Figuur 4: (a) A graphic representation of a robot in a simple environment and (b) the method to calculate the angles from data points.

2.5 Descriptors

Once the features are extracted from the data we want to describe the recording in order to match them with each other later on. To describe what is seen by the sensor we will use descriptors. A descriptor is created for all the recordings that are made. These descriptors can then be compared with each other to establish their agreement. The idea of using descriptors and matching them came from [7]. They create descriptors for images and compare them in very different ways than we do in our research. However the basic idea about descriptors and matching them came from that article

The descriptor we will create for a recording is a string that represents all the features that are



Figuur 5: Graphical representation of the data of an outer corner taken from position nr 5, with the features: line pieces (black to yellow dot), inner corners (red dots) and outer corners (green dots).

detected in that particular recording. The characters in the string each have their own meaning. We have used the L for a line piece feature, an I for an inner corner feature and an O for an outer corner feature. An example of a descriptor is the string LLLILLLL, which is a combination of the above characters and is created from the recording of an inner corner. To optimize the probability of matches between recordings, we simplify the descriptors by removing the consecutive occurrences of the line piece features. The descriptor from the example above LLLILLLL will then become LIL. In a simplified descriptor the exact location of the feature is not seen anymore between the series of Ls. Therefore the simplified descriptors ensure that the features will get more viewpoint invariance and are more robust. This also aids in the noise insensitivity of the features.

2.6 Matching with Levenshtein distance

We would like to compare the different recordings of the different features with each other to see what their match quality is. When the descriptors are created they will be matched using Levenshtein distance [8]. This measure of comparison says how many characters need to be transformed, added or deleted in order to turn one string into another. For every action the Levenshtein distance will increase with one. This means that a perfect match has a Levenshtein distance of 0, since there is nothing to be done to turn one string into the other. Among others we use the Levenshtein distance as a measure of error to help the features get more match quality.

3 Results

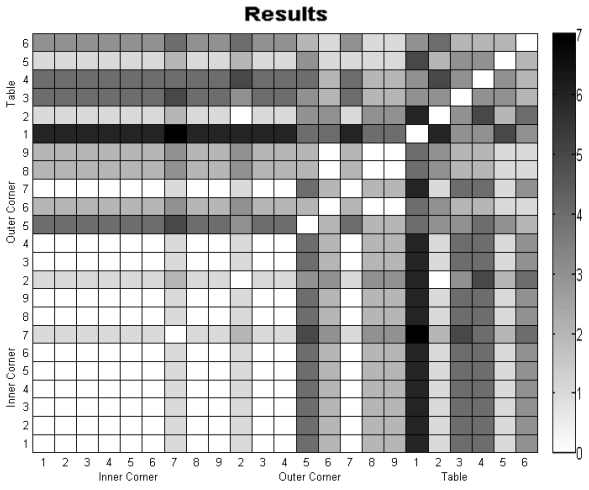
We have compared all of the 23 recordings, which spread over the inner corner, the outer corner and the table, with all of themselves. As said in the method section the recordings are compared with each other by means of the Levenshtein distance. We will obtain a matrix of 23-by-23 results of the comparisons.

The following figure 6 shows a pseudocolor (checkerboard) plot of the results of this research. This figure was made in a gray scale and how darker the color is, how bigger the Levenshtein distance was. Places with a white color are comparisons that came out with a Levenshtein distance of zero and are therefore perfect matches.

The biggest Levenshtein distance measured was 7 and the smallest 0. The plot shows that mostly all of the recordings of the inner corner match perfectly with the other recordings of the inner corner. They also match perfect with some of the recordings of the outer corner. So the inner corner does match well with the outer corner in some degree. The inner and outer corner both do not seem to match well with the table feature. The outer corner does not match very well with different recordings of the same feature. Recordings of the table do match even worse with different recordings of the same feature. The plot also shows a diagonal of perfect matches on all of the recordings. This diagonal of perfect outcomes results from the comparison of the recordings with themselves.

4 Discussion

From the results we can conclude several things and say some things about this research. First of all, the recordings of the inner corner match almost perfectly with every other recording of the inner corner. These are the results we were looking for, since we wanted to extract features, which

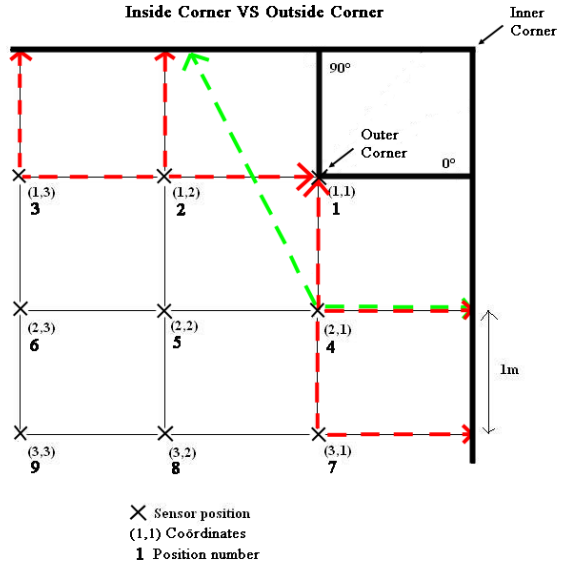


Figuur 6: Pseudocolor (checkerboard) plot in gray scale of the results of comparing every recording with each other.

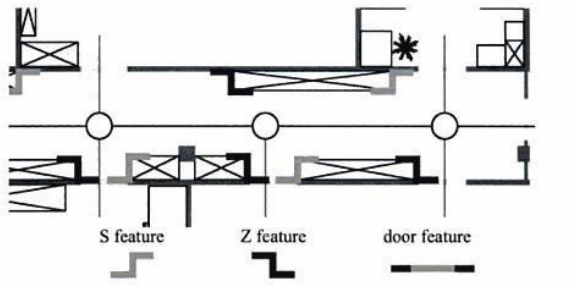
among others posses viewpoint invariance, from the environment. The causes of the good matches are the facts that the descriptors are simplified, as discussed in the method section, and the use of Levenshtein distance to measure the error between descriptors. The simplifying of the descriptors causes the descriptors of the inner corner recordings to almost all become LIL and therefore match perfect with the other recordings that have the same simplified descriptor. The Levenshtein distance does not take in account what kind of transformation is used to turn the one string into the other. For example, the distance between the unsimplified descriptors LILLL and LLLIL is 2, but the distance between LILLL and LLILL is also 2. In this procedure some information about the exact location of the feature within the descriptor is lost. But that is exactly what we would like to see, since we want our features to be robust and posses viewpoint invariance. When the features are recognized and well matched, we can use information from the angle at which the feature has been seen to compute the location of the robot in the environment. Correct self localization however, does require the detection of two corner features and an internal map of the environment. In that case you can combine the detected features with the angles at which they are seen and with each other. Looking at the inner corner feature we can say that the identification of the feature goes well, and will aid in self localization.

The second conclusion we can make from looking at the results, is that the recordings of the outer corner match to some degree with each other. We do not see such a good matches as with the inner corner, but a couple of recordings match perfect with the other. This means that the outer corner feature can be used for self localization but is much more unreliable than the inner corner feature. The reason that the outer corner feature does not perform as well as the inner corner feature, is that it consist of two inner corner features and one outer corner feature, as discussed in the method section. This unlike a true outer corner feature which consists of just one outer corner feature. The reason we have not used such a true outer corner feature to record, is that such a feature was not to be found in the laboratory in which we did our data recording. Therefore we had to construct an outer corner feature ourselves. When looking at the results of the matches between the recordings of the table, we see that they all have a quite high Levenshtein distance and also there are no perfect matches. Therefore we can conclude that the table feature is not a good feature to work with for self localization in this research.An occurrence that can be seen in the results which we would like to explain is that some recordings of the outer corner match perfectly with almost all of the recordings of the inner corner. There is a logic explanation for that occurrence. The recordings of the outer corner from position 2, 3, 4 and 7 are quite similar to the recordings of the inner corner from position 1, 2, 1 and 4 respectively. All the recordings are done under a 90 degree field of view. As shown in figure 7, the recordings of the outer corner, which are marked with a red dotted line, look like a regular inner corner from the earlier named positions. That is the reason why the matches in the results occur. A possible solution to this problem is to make 360 degree recordings in stead of 90 degree recordings and take the relevant data out of those recordings. This way you can look at bigger parts of the environment if necessary and extract more difficult features and than maybe solved the problem discussed above. The dotted green line in figure 7 shows such a bigger recording of the environment.

Something we should ask ourselves is whether the measurement with Levenshtein distance is precise enough for this research. As discussed above it does not take in account some information, like the distance between several characters in a descriptor (an I and L, and an I and O) and the distance between the same feature within a string (the LILLL example in the beginning of this section). But since we do not need that kind information in this research, the Levenshtein distance was a proper type of measurement. However, that does not mean that it is suitable for any type of research since some information is lost. Researchers using the Levenshtein distance should be careful and consider their choice very well. With our method of looking at environments and forming descriptors we should also be able to detect other kinds of features. An example of these recognizable features are the Z- and S-corners, which consist of the combination of an in- and outside corner and an out- and inside corner respectively. An example of those features is given in figure 8 which is taken from book [6]. In that research they have tried to detect those combination-features.



Figuur 7: Graphical overview of the 90 degree sight from positions 2, 3, 4 and 7 (red dotted lines) and a bigger than 90 degrees recording (green dotted lines).



Figuur 8: Multiple geometric features in a single hallway, including S- and Z features.

In our research if you look closely at the outer corner, you can see that the Z- and S-corners are hidden in that environment. These features are also recognizable within the descriptors of the environment. Our method should in theory also be able to detect those features, but we have decided that that was beyond the scope of this research. But we advise other researchers using our research, to include those combination features, because they can be quite informational for self localization.

There is also data recorded of the temperature of the environment at the different angles, but this data is unused in this research. Temperature readings however can be useful in the future to detect living objects in the environment.

Concluding we can say that a portion of our results were very good and confirmed our expectations. The facts that certain features could not as well be used for self localization as others was unfortunate, but the conclusions about those features were useful. With our method we are able to extract and describe certain simple features and structures. And as said before, those features, in combination with their angle and an internal map, can aid in self localization of autonomous (vacuum cleaning) robots.. With some more time and methods to create and explore our dataset, we could refine this research and achieve even better results.

5 References

[1] Sharp Microelectronics. GP2Y0A710K0F IR Distance sensor. Datasheet on http://www.sharpsma.com/Page.aspx/americas/en/part/GP2Y0A710K0F/

[2] Johannes Feulner, Jochen Penne, Eva Kollorz, Joachim Hornegger, Robust Real-Time 3D Modeling of Static Scenes Using Solely a Time-of-Flight Sensor, 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Miami, FL, USA, June 2009, pp. 74-81.

[3] Chang hwan Lee, Woong keun Hyun en Kwang taek Kim, A simultaneous map building system by using developed photo PSD sensors, SICE-ICASE International Joint Conference 2006, Bexco, Korea, October 2006, pp. 2999-3004.

[**4**] Kurt Konolige, Jospeh Augenbraun, Nick Donaldson, Charles Fiebig en Pankaj Shah, A low-cost laser distance sensor, 2008 International Conference on Robotics and Automation, Pasadena, CA, USA, May 2008, pp. 3002-3008.

[**5**] Heon-Hui Kim, Yun-Su Ha, Gang-Gyoo Jin, A Study on the Environmental Map Building for a Mobile Robot Using Infrared Range-finder Sensors, Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, October 2003, pp. 711-716.

[**6**] Roland Siegwart, Illah R. Nourbakhsh, Introduction to Autonomous Mobile Robots, The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2004. Massachusetts, 2004, pp. 154-163.

[**7**] Krystian Mikolajczyk, Cordelia Schmid, A performance evaluation of local descriptors, IEEE Transactions on pattern analysis and machine intelligence, October 2005, vol. 27, no. 10, pp. 1615-1630.

[**8**] Charlotte Gooskens, Wilbert Heeringa, Perceptive Evaluation of Levenshtein Dialect Distance Measurements Using Norwegian Dialect Data, 2004, Language variation and Change, vol 16, no.3.